

SIEMENS



Desigo™ TX Open **TX Modbus** **Engineering guide**

Version 6.3 and later

Contents

1	About this document	3
1.1	Revision history	3
1.2	Reference documents	3
1.3	Before you start	3
1.3.1	Trademarks	3
1.3.2	Copyright	4
1.3.3	Quality assurance	4
1.3.4	Cyber security disclaimer	4
1.3.5	Document use / request to the reader	5
1.4	Target readers	5
1.5	Glossary of new terms, abbreviations	6
2	Introduction	7
2.1	Integration of the TX Open Modbus Solution	7
2.2	LED indication	9
2.3	Functional scope and limitations to Modbus standard	10
2.3.1	Emergency position value/default value	11
2.4	Data types and mapping data to function blocks	11
2.4.1	Modbus device parameters	12
2.4.2	Modbus point parameters	13
2.5	Example: Create an IO Open Point Template for Modbus	14
3	Modbus protocol application functionality	16
3.1	Overview	16
3.2	Communications	17
3.2.1	The Modbus protocol	17
3.2.2	Poll response of the Modbus protocol application	18
3.2.3	Updating the output values from the viewpoint of the Modbus master ...	19
3.2.4	Collating several data points in a single telegram	19
3.2.5	Suppression of alarm messages in the event of a bus failure	20
3.3	Parameters for Modbus devices and communication	21
3.4	Parameters for Modbus points	23
3.4.1	Data types and mapping data to function blocks	25
3.4.2	Detailed description of the mapping information	26
3.5	Extensions to the Modbus protocol	33
3.5.1	Transmit 32-bit and 64-bit values	33
3.5.2	BitPos function: Read/Write individual bits and multiple bits in the Modbus registers	34
3.5.3	Suppress transmission of a variable number of data points	35
3.5.4	Feedback Function and value sequences	35
3.5.5	Periodic writing of outputs	36
3.5.6	Format conversion and value shifting	37
3.5.7	Grouping of output registers	37
3.5.8	BACnet reliability	38
3.5.9	Broadcast telegrams	38
4	Create protocol applications for projects (by an RC)	39
5	Technical data	39
6	SW and FW versions	39

1 About this document

1.1 Revision history

Version	Date	Changes	Section	Pages
_09	Nov 2024	Minor corrections V6.3	All	
_08	August 2018	Minor corrections V6.1	All	
_07	2018-09-05	Minor corrections	-	
_06	2017-03-31	Changes for Version 6.1 and minor corrections	All	
_05	2016-07-31	Changes for Version 6.0 and minor corrections	All	
_05	2014-08-30	Changes for Version 5.1 and minor corrections	3.5.7	
_04	2010-03-30	Changes for Version 4.1 and minor corrections	2.4, 3	
_03	2009-11-30	Minor corrections	3.2.4, 3.3, 3.4.2, 3.5.2	
_02	2009-07-03	Minor corrections	2.1,	7, 13

1.2 Reference documents

Ref.	Document title	Doc. type	Doc. no.
[1]	TX Open module (TXI1.OPEN)	Data sheet	CM2N8185
[2]	TX Open module (TXI2.OPEN)	Data sheet	CM2N8187
[2]	TX Open Workflows	Quick reference Guide	A6V10963119

1.3 Before you start

1.3.1 Trademarks

The trademarks used in this document and their lawful owners are listed in the table below. The use of these trademarks is subject to international and national statutory provisions.

Trademarks	Legal owner
BACnet	American National Standard (ANSI/ASHRAE 135-1995)
Microsoft ...	Microsoft Corporation; see https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks
MODBUS®	The Modbus Organization, Hopkinton, MA, USA
Windows ...	Microsoft Corporation
WindowsCE	Microsoft Corporation

All products listed in the table are registered (®) or unregistered (™) trademarks of the corresponding owner shown in the table. In view of this information, additional markings (such as the use of symbols like ® and ™) for the trademarks are omitted in the interest of greater legibility.

1.3.2 Copyright

Reproduction and distribution of this document is only permitted with the express permission of Siemens and only for authorized persons / enterprises with appropriate technical expertise.

1.3.3 Quality assurance

These documents were prepared with great care.

- The contents of all documents are checked at regular intervals.
- Any corrections necessary are included in subsequent versions.
- Documents are automatically amended as a consequence of modifications and corrections to the products described.

Please make sure that you are aware of the latest document revision date.

If you find lack of clarity while using this document, or if you have any criticisms or suggestions, please contact the product manager in your nearest branch office.

The addresses of the Siemens regional companies are available at

<http://www.siemens.com/sbt>

1.3.4 Cyber security disclaimer

Siemens products and solutions provide security functions to ensure the secure operation of building comfort, fire safety, security management and physical security systems. The security functions on these products and solutions are important components of a comprehensive security concept.

It is, however, necessary to implement and maintain a comprehensive, state-of-the-art security concept that is customized to individual security needs. Such a security concept may result in additional site-specific preventive action to ensure that the building comfort, fire safety, security management or physical security system for your site are operated in a secure manner. These measures may include, but are not limited to, separating networks, physically protecting system components, user awareness programs, defense in depth, etc.

For additional information on building technology security and our offerings, contact your Siemens sales or project department. We strongly recommend customers to follow our security advisories, which provide information on the latest security threats, patches and other mitigation measures.

<http://www.siemens.com/cert/en/cert-security-advisories.htm>

1.3.5 Document use / request to the reader

Before using our products, it is important that you read the documents supplied with or ordered at the same time as the products (equipment, applications, tools etc.) carefully and in full.

We assume that persons using our products and documents are authorized and trained appropriately and have the technical knowledge required to use our products as intended.

More information on the products and applications is available:

- On the intranet (Siemens employees only) at <https://siemens.sharepoint.com/teams/P0002723/SitePages/Home.aspx>
- From the Siemens branch office near you <http://www.siemens.com/sbt> or from your system supplier
- From the support team at headquarters fieldsupport-zug.ch.sbt@siemens.com if there is no local point of contact

Siemens assumes no liability to the extent allowed under the law for any losses resulting from a failure to comply with the aforementioned points or for improper compliance of the same.

1.4 Target readers

The document describes TX Open commissioning. There are three main workflows to engineer, monitor and control using TX Open modules. For more information about the steps and functions of TX Open tool refer to the online help.

1.5 Glossary of new terms, abbreviations

This table contains important terms (not all inclusive), where, to some extent, outdated terms are used in the referenced documents.

Term	Explanation
Modbus	Modbus protocol is a communications protocol based on master/slave or client / server architecture transmitted on a serial interface (standards: EIA-RS-232, EIA-RS485).
Desigo	Siemens building automation and control system.
Automation stations (AS)	Controller for the Desigo system (PXC100, PXC200).
TX-I/O	I/O modules product range for the Desigo system.
Island bus	Field bus for TX-I/O.
TXI1.OPEN	ASN for the HW product TX Open, island bus device, device for the TX-I/O product range with limited amount of data points (100)
TXI2.OPEN	ASN for the HW product TX Open, island bus device, device for the TX-I/O product range with limited amount of data points (160)
TXI2-S.OPEN	ASN for the HW product TX Open, island bus device, device for the TX-I/O product range with limited amount of data points (40)
Protocol application	Software for TXIx.OPEN, that processes a specific protocol on the serial bus. Can be uploaded on-site to TXIx.OPEN. The document describes protocol application properties for Modbus devices.
XWP	XWORKS plus, Desigo system toolset (as of Version 4 and later).
Point Configurator	XWP component to generate data points and assign data points to the Desigo system components, among others for TXIx.OPEN and data points for TXIx.OPEN as well.
IOOCD	IO OPEN Configuration Description = data exchange file for configuration data for the TXIx.OPEN device.
IOOPT	IO OPEN Point Template = data exchange file for a configuration that describes a specific third-party device.
IOMD+	Internal data file used to download to the TXIx.OPEN device (content = required data from IOOCD, XML code).
RS-232	Standardized serial interface, EIA-RS-232.
RS-485	Standardized serial interface, EIA-RS-485.
ASCII (Modbus)	Operating mode for data transmission: Text characters legible for people are transmitted.
RTU	Remote Terminal Unit = Operating mode for Modbus data transmission. Binary encryption of transmitted data.
VSD	Variable Speed Drive.
VISUAL C++	Programming language for generating protocol applications.

2 Introduction

2.1 Integration of the TX Open Modbus Solution

10571Z01

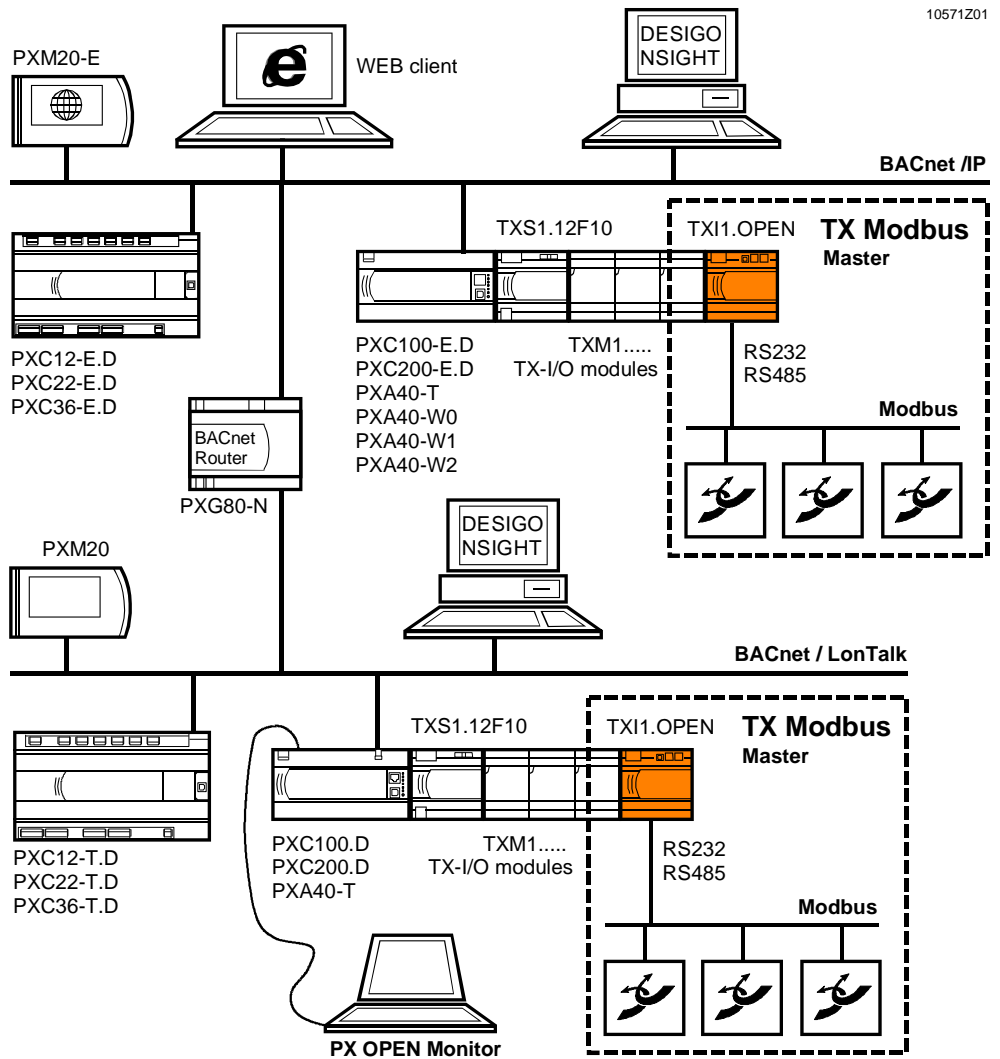


Figure 2–1: TXI1.Open in the Desigo PXC product range



Figure 2–2: TXI2.OPEN Hardware

Modbus devices are integrated into the Desigo system using the TXIx.OPEN. TX Open is a package of tools, workflows and files to configure and commission with Desigo.

TXIx.OPEN is loaded with protocol applications for Modbus and then operates as Modbus master.

The values of the Modbus data points and the status of the existing data connection with the data points are transmitted to the automation station (AS) via the island bus and mapped to BACnet objects in the automation station using duplex transmission. This way, Modbus data points can be made available to all the devices and applications in the Desigo system.

An operator interface is available in the TXIx.OPEN tool to monitor and analyze data traffic on the Modbus for diagnostics, value display and reading of versions.

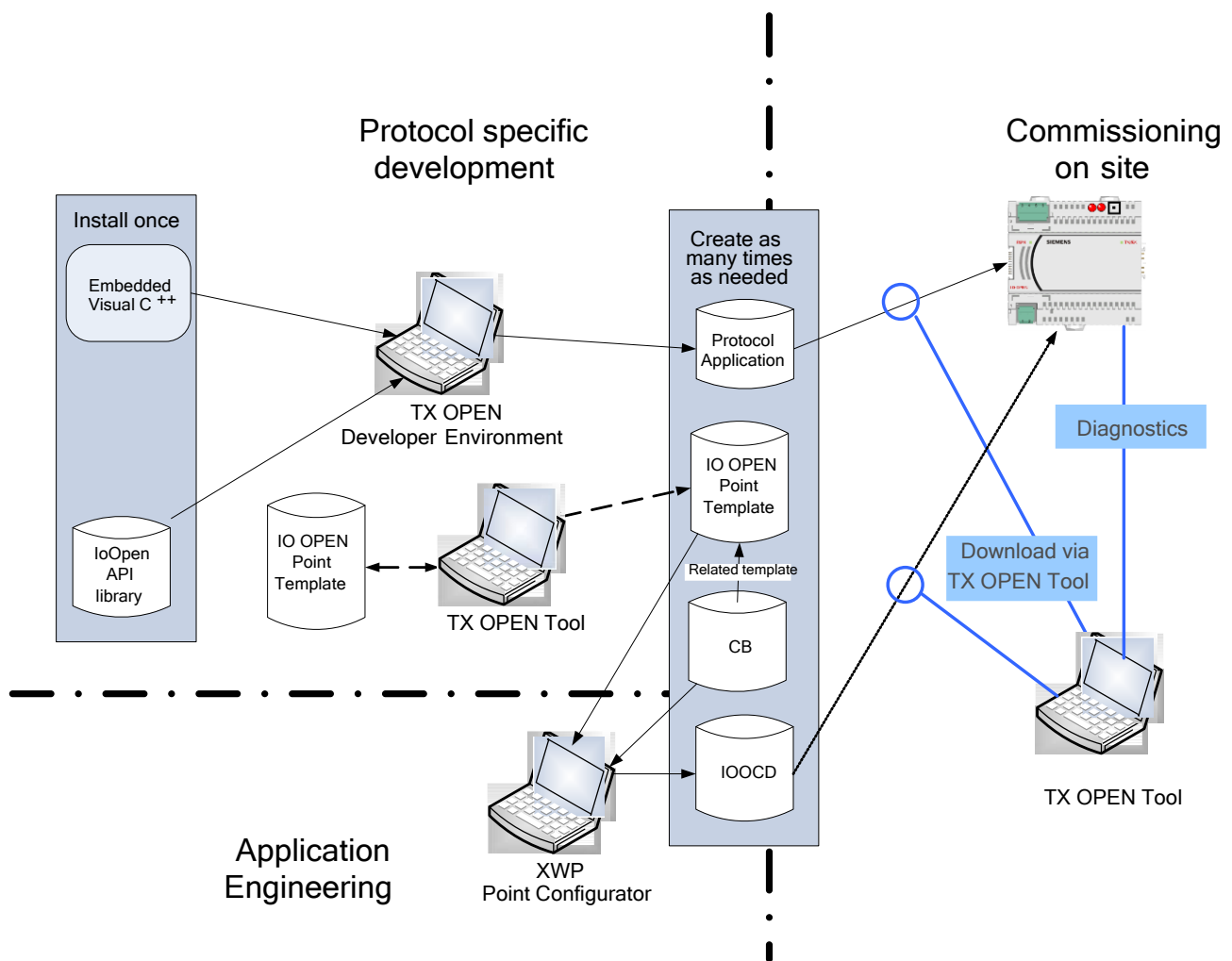


Figure 2-3: Overview of TX Open

2.2 LED indication

The LEDs indicate activities on the island bus and on the serial interface (one color each for receive and transmit).

Normal state

RUN (green)	The LED indicates whether island bus communication is operating. Constant green means that communications are running.
-------------	--

COM (green/red)	The LED indicates communication on the RS side. Red = TX (transmit); green = RX (receive). Error: Only TXIx.OPEN sends if the color does not change. No devices respond. The color is then stable or flashes red.
-----------------	---

Error state

RUN (green)	The LED flashes in the event of an error. No indication (neither flashing nor lit) indicates that the device is only operating to a limited extent.
-------------	--

COM (green/red)	Continuous red indicates an error. Flashing red without intermittent short, green flashing indicates that the device is operating in principle but none of the connected devices are responding. This occurs, for example, when communication is not set up correctly (such as an incorrect baud rate, incorrect parity, etc.).
-----------------	---

2.3 Functional scope and limitations to Modbus standard

Functional scope:

Software options:

- Use as Modbus master
- Transfer modus: ASCII and RTU.
- Modbus TCP supported

Hardware option:

- RS232
- RS485 bus connection: with / without bus connection.
Bus connection = termination resistor (120 Ohm).
- RJ45 connector for use of Modbus TCP integration

Table 2–4: Modbus function codes supported

Function code	Description	Multiple values in a telegram
01	Read coil status	Yes
02	Read input status	Yes
03	Read holding register	Yes
04	Read input registers	Yes
05	Force single coil	
06	Preset single register	
15	Force multiple coils	Yes
16	Preset multiple register	Yes

Limitations

Cannot be used as Modbus slave in the described protocol application.

Maximum of 160/40 points from Modbus as the sum total of all devices.

Recommended 3-20 per Modbus device depending on application.

Extreme values:

- 100/160 for exactly 1 Modbus device.
- Maximum of 160/40 Modbus devices with one data point each.

Multi-drop mode is only possible to a limited extent. We recommend operating the network in the ASCII mode for multi-drop.

Only one Modbus device per TX Open modules may communicate when operating the Modbus devices via RS-232.

You can integrate a point template up to 100 data points for TXI1.OPEN module and up to 160 data points for TXI2.OPEN module.

For the new module TXI2-S.OPEN a maximum of 40 data points is allowed.

2.3.1 Emergency position value/default value

The emergency positioning value is only used for OUT points and sent to the third-party device when the island bus master fails.

The emergency positioning value is defined in Desigo with BackupValue and BackupMode on the point and transmitted via the island bus to the TX Open module.

TX Open tool has a data field (DefaultValue) defining the emergency positioning value issued as long as no emergency positioning value is configured via the island bus.

The output of an emergency positioning value upon startup is suppressed.

The default value may nevertheless be used, if the following functions are used:

- Output of value with MappingInfo3 (BitPos 1-16).
- Output of the value is forced with function /R1,/R2 or /R3 (see 3.5.5).

2.4 Data types and mapping data to function blocks

Supported function blocks on the Desigo side: AI, AO, DI, DO, MI, MO, CI.

Note: These are resource types in V4. There sometimes are multiple types of function blocks per resource type, for example, function blocks CI and ACC for resource type CI.

Supported data types in the Modbus device:

For 1 register with 16 bit: BOOL, UINT16, SINT16.

For 2 registers with 16 bits each: UINT32, SINT32, FLOAT.

For 4 registers with 16 bits each: DOUBLE (*from version 4.1*).

Note: The function MappingInfo3 can be used to read out an individual bit from a register.

Mapping steps from Modbus to Desigo blocks (Details see Section) **3**

1. On the Modbus, bits or 16-bit registers are read /written.
2. For FLOAT, UINT32 and SIN32, 2 back-to-back 16-bit registers are read /written – for DOUBLE: 4 registers.
3. The Modbus protocol application maps these bits and registers to the island bus I/O data types as determined by the function block: (AI, AO=F32; CI=U32; MI, MO=U16; DI, DI=U8). Basic rules and MappingInfo3 and MappingInfo4 determine the mapping (See Section 3 for details).
4. The island bus type formed by the Modbus protocol application is transmitted as raw value to the function blocks via island bus (corrected with +1 / -1 for MI / MO).
5. Additional processing functions are available in the function blocks depending on the block (e.g. Slope/Intercept).

Table 2–5: Recommended mapping of Modbus data types to Desigo block types:

Desigo Resource Type	Island bus	Modbus data type (third-party signal type)					
		BOOL	UINT16	SINT16	UINT32	SINT32	FLOAT
AI	F32, IEEE FLOAT	OK	OK	OK	OK (accuracy loss)	OK (accuracy loss)	OK (accuracy loss for DOUBLE)
AO	F32, IEEE FLOAT						OK (accuracy loss for DOUBLE)
DI	U8 0=false, 1=true	OK					
DO	U8 0=false, 1=true	OK	OK	OK	OK	OK	OK
MI	U16	OK	OK	OK			
MO	U16		OK		OK	OK	OK
CI	U32	OK	OK		OK		

FLOAT and DOUBLE are IEEE Float numbers with 32 / 64 bit respectively.

2.4.1 Modbus device parameters

Modbus device and configuration parameters

Summary of the most important communication parameters for the Modbus itself and for Modbus devices. See Section 3 for details.

- 3rd DevAddr include a number.
- 3rd BusType RS232, RS485, RS485T (with terminating resistor), IPv4.

Assignments in the **Communication Settings** field is:

Baudrate, Parity, Length, Stopbit, BusType, 0, 0, Transfermode, FreeText

Table 2–6: Setting parameters for Communication Settings

Type	Input field	Description	Example
BusType	BusType	"RS232" "RS485" "RS485T" are options for Modbus on RS232 or RS485. Especially for RS485 is the option without / with bus connection. For Modbus TCP use "IPv4"	RS232
Baudrate	Commsettings	Baud rate on Modbus: 300 / 600 / 1200 / 2400 / 4800 / 9600 / 19200 / 38400 / 57600 / 115200	9600
Parity	Commsettings	Parity on the Modbus: N: No parity; E: Even; O: Odd M: Mark parity; S: Space parity.	N
Length	CommSettings	Number of data bits per character on the Modbus: 7 / 8	8
StopBit	CommSettings	Number of stop bits on the Modbus: 1 / 1.5 / 2	1
TransferMode	CommSettings	Transfer mode: "RTU" "ASCII" "TCP"	ASCII

2.4.2 Modbus point parameters

Modbus point addressing

For each point, two fields 3rd Conversion Set and 3rd Signal Type determine the address used to read on the Modbus device that reads data from this address (bit, register with 2 bytes, 2 back-to-back registers with a total of 4 bytes) and how they are converted to a Desigo data type. Field display:

3rd ConversionSet:

MI1;MI2;MI3;MI4 Text with 4 numeric values (MappingInfo) separated by;

3rd SignalType

Text Data type and optional supplemental information (one or more /parameters).

Table 2–7: Overview of setting parameters for points

Description	Description	Example (notation from IOCD)
One Modbus function code per point (MappingInfo1)	Function code with which the data point is to be transmitted. See overview table in Section 2 and all details in Section 3.	ThirdConversionSet="3;7;0;0". ↑
One Modbus register address per point (MappingInfo2)	Modbus register address of the data point between 0 . 65535. For values transmitted in two registers, the register with the lower value must be entered here.	ThirdConversionSet="3;7;0;0". ↑
One supplemental function per point (MappingInfo3)	0=No function. Refer to Section 3 for functions.	ThirdConversionSet="3;7;0;0". ↑
One format instruction per point (MappingInfo4)	0=No function. Refer to Section 3 for functions.	ThirdConversionSet="3;7;0;0". ↑
One data type per point.	Data type in the Modbus device. BOOL, UINT16, SINT16, UINT32, SINT32, FLOAT, DOUBLE	ThirdSignalType="UINT16".

All the details on setting parameters for points available in Section 3.

2.5 Example: Create an IO Open Point Template for Modbus

This section is a summary from a third-party data sheet for a particle counter and its implementation and storage in the Point Template (IOOPT) and the configuration file (IOOCD).

Third-party device description

Device functionality for the Remote Airborne particle counter

The device is a remote airborne particle counter for inert gas samples in clean rooms. It collects data for the plant monitoring system. System integration is physical via a RJ-45 connection and logical, via RS-485 Modbus communication.

Modbus functionality

Modbus command function codes

Description	Transmitted code	Modbus code
Read Registers	0x03	3
Preset Single Register	0x06	6
Who Is There?	0xff	Not Modbus

Function code **3** (read registers):

- Telegram creates 3 as function code, start register address, number of register, CRC.
- Read telegram with 3 as response code, 2 bytes per requested register, CRC.

Function code **6** (preset single register):

- Telegram creates with 6 as function code, register address, register default data, CRC.
- Read telegram with 6 as function code, register address, register default data, CRC.

Command register (16-bit wide)

- Address 0: Control
Device controllable by only one register; Operation Control Code Register with command list:
1 = Start count; 2 = Stop count/laser off; 3 = Set alarm;
4 = Delete alarm; 5 = NIU; 6 = Laser on; 7 = Laser off.

Read Registers (16-bit wide).

- Address 0: Status
Bit 1 = Count On/Off; bit 2 = Laser On/Off; bit 3 = Alarm On/Off;
Bit 4 = CAL Yes/No.
- Address 1: Ch1 size label bytes 1,2.
- Address 2: Ch1 size label bytes 3,4.
- Address 3: Ch1 counts upper word.
- Address 4: Ch1 counts lower word.
- Address 5: Ch2 size label bytes 1,2.
- Address 6: Ch2 size label bytes 3,4.
- Address 7: Ch2 counts upper word.
- Address 8: Ch2 counts lower word.
- Address 9 Message count.
- Address 10: Address Msg Count; Message count up to this address.
- Address 11: Good Msg Count; count of good messages up to this address.
- Address 12: Unknown Cmd Count.
- Address 13: Unknown Error Count.

- Address 14: Unknown Error Count; count of "message overrun errors".
- Address 15: Noise Error Count; count of message errors caused by noise.
- Address 16: Framing Error Count; count of messages with framing error.
- Address 17: Parity Error Count; count of messages with parity errors.
- Addresses 18-33: Version date; bytes 1,2; 3,4; ...; 31,32.
- Addresses 34...41: Model name; bytes 1,2; 3,4; ...; 15,16.

Create template in the TX Open tool

The IO Open Point Template is created in the TX-I/O Open tool using the data for the Modbus device.

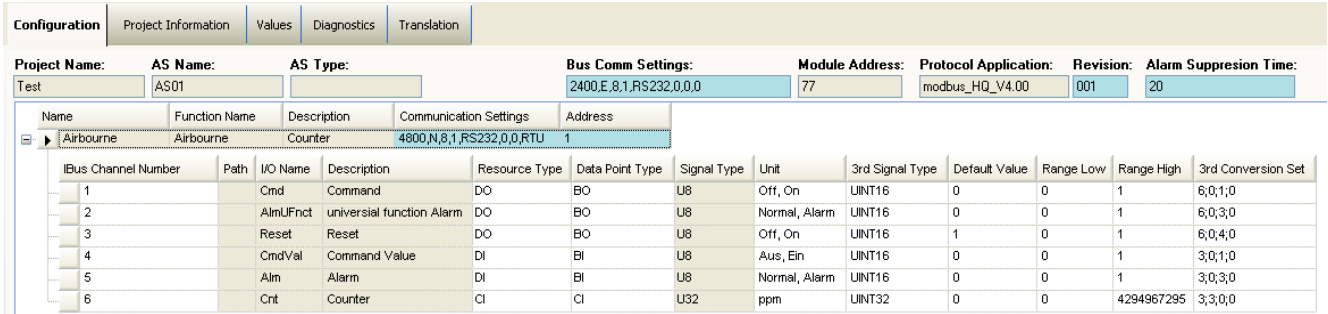


Figure 2–7: TX Open tool: IOOPT for particle counter

In the example, only a few, truly relevant data points are transferred to the template. This is typical for these types of devices: Lots of points that are needed for special commissioning and just a few points that are truly used in the operation of the superposed system. For templates reused in the libraries, all points must be added to the template, since the only way to add missing points in the template in the workflow is by creating a new template; conversely deselecting unused points is simple.

The BitSet functions in the MappingInfo3 are used for direct access to individual bits in status and command. A somewhat more comfortable Modbus device would offer the function codes 2 (Read Input Status) and 5 (Force Single Coil) and provide a separate address for each bit.

3 Modbus protocol application functionality

3.1 Overview

- The Modbus protocol application integrates data points from Modbus devices into the Desigo system.
- The values of the Modbus data points and the status of the existing data connection with the data point are mapped to BACnet objects by the Modbus protocol application.

Serial transmission modes:

RTU mode or ASCII mode.

Supported Modbus function codes

Function code	Modbus function	Multiple values in a telegram
01	Read coil status	Yes.
02	Read input status	Yes.
03	Read holding register	Yes.
04	Read input registers	Yes.
05	Force single coil	No
06	Preset single register	No
15	Force multiple coils	Yes.
16	Preset multiple register	Yes.

3.2 Communications

3.2.1 The Modbus protocol

The following section provides only a brief overview of the Modbus protocol. For the full specification, refer to "Modicon Modbus Protocol Reference Guide www.modbustools.com/PI_MBUS_300.pdf". <http://www.modbus.org> is also a source of information on Modbus.

Master/slave protocol

The Modbus is a master/slave protocol. By definition, this means that a Modbus network contains one, and only one, master and at least one slave.

Transactions on the Modbus

The Modbus master starts the transactions on the network with a slave query. The slave either responds positively with the requested service (*response*) or transmits an "exception message". Query and response sequences consist of 2 Modbus telegrams.

Function codes

The type of transaction is defined by the function code transmitted in the Query Modbus telegram. A function code defines the following:

- Structure of the telegram, query and response.
- Direction of data transmission (Master → Slave or Slave → Master).
- Data format of data point (bit or 16-bit register).

Modbus function code (MappingInfo1)	Modbus-storage range (Coil/Register Numbers)	Description of the Modbus function	Function blocks in CFC Editor	Application
01	0X	Read coil status	DI	Read one or more DIs.
02	1X	Read input status	DI	Read one or more DIs.
03	4X	Read holding register	AI, MI, CI	Read one or more AIs or MIs or CIs.
04	3X	Read input registers	AI, MI, CI	Read one or more AIs or MIs or CIs.
05	0X	Force single coil	DO	Write single DO.
06	4X	Preset single holding register	AO, MO	Write an AO or MO.
15 = Hex 0F	0X	Force multiple coils	DO	Write one or more DOs.
16 = Hex 10	4X	Preset multiple holding registers	AO, MO	Write one or more AOs or MOs.

Transmission modes

The Modbus protocol defines two alternative serial transmission modes: The two modes have the following characteristics:

RTU (Remote Terminal Unit) mode

- Binary-coded data.
- Start and end of telegrams marked by timed pauses (a "silent interval") between the characters transmitted.
- Checksum algorithm: CRC (Cyclic Redundancy Check).

ASCII mode

- Data in hexadecimal notation.
- Beginning and end of telegrams marked by start and end characters.
- Checksum algorithm: LRC (Longitudinal Redundancy Check).
- Modbus ASCII-Mode is supported.

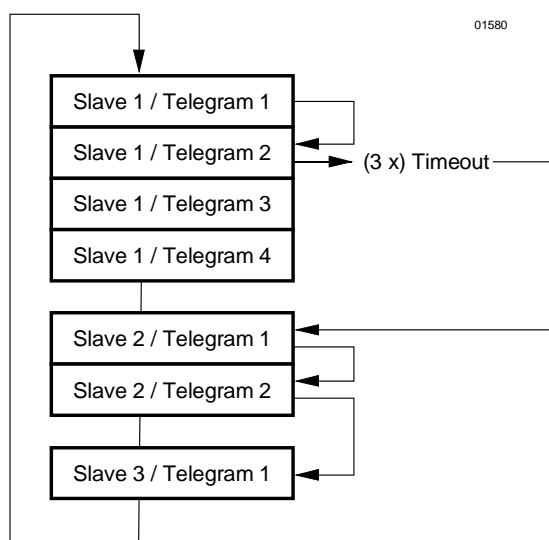
3.2.2 Poll response of the Modbus protocol application

The Modbus master reads the data points from the slaves at regular intervals, using the relevant function code (01, 02, 03 or 04). If a slave fails to respond, the master waits after each query for the (relatively long) response timeout time to elapse. This causes a major delay in the exchange of data with those slaves that still must respond.

To overcome this, the Modbus protocol application master mode as follows:

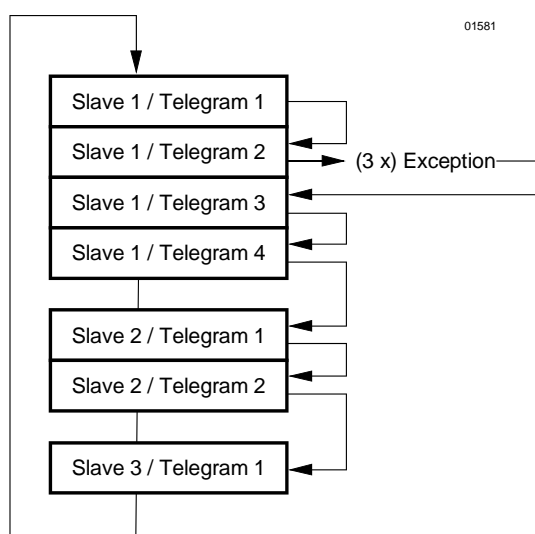
A) Slave fails to respond to query from master.

- Telegram is repeated.
- If the slave still fails to respond, any pending queries (read or write) for this slave are skipped until all the pending queries for the other slaves have been processed once.



B) Slave responds to a query by sending an exception message

- Telegram is repeated.
- If the slave continues to respond with an exception message, the next pending query is transmitted.



Note

The state of the Modbus data points is mapped to the "Reliability" parameter, Rlb (refer to BACnet Reliability).

The sequence of telegram transmission from master to slave can be influenced via parameters FrTmo, ChTmo, /SD and /TD, see section 3.3.
/SD and /TD are only supported in Version 4.1 and later.

3.2.3 Updating the output values from the viewpoint of the Modbus master

Function Codes 05, 06, 15 and 16 serve to write data point values to the Modbus slave. These values are written once when the master is started (with a positive acknowledgement) and are subsequently rewritten only in the event of a change in the data point value.

There is a risk of data inconsistency after a reset or after a replacement of the Modbus slave device, since the value update is initiated only in the master in the event of a change of value.

To overcome this, the Modbus protocol application uses the following mechanism when operating as master: /R parameters per point.

/R15 does only work properly in case the MODBus slave is once connected successfully.

3.2.4 Collating several data points in a single telegram

Modbus Function Codes 01 (Read Coil), Codes 02 (Read Input Status), 03 (Read Holding Register), 04 (Read Input Register), 15 (Force Multiple Coils) and 16 (Preset Multiple Register) can be used for the combined transmission of a variable number of Modbus data points in a single Modbus telegram.

Modbus master

Function Codes 01, 02, 03 and 04 are used to read multiple data point values from the Modbus slaves. In other words, these values are polled at regular intervals in the slaves.

The Modbus protocol application automatically compiles these poll telegrams to allow the largest possible number of data points to be transmitted in a telegram. Compiled transmission is only done with adjacent registers of the same data type (DataType, see section 3.4).

Version 4.1 and later: the maximum number of collected data can be limited via parameter "L=number". "Number" indicates the maximum number of bytes in the telegram.

Function Codes 15 and 16 serve to write multiple data point values to the Modbus slave. These values are written once when the program starts, and subsequently whenever the data point value changes.

The Modbus protocol application automatically compiles these write telegrams to allow the largest possible number of changed data points to be transmitted in a telegram. Compiled transmission is only done with adjacent registers of the same data type (DataType, see section 3.4).

Notes

- As only the address of the first data point and the total number of data points can be inserted in the Modbus telegram, it is only possible to combine data points with addresses in continuous ascending order.
- /S forces transmitting each point in a separate telegram (see section 3.5.3).

Response in the event of engineering errors

If the Modbus master asks a slave for compiled data points and this is refused by the slave, the master does not stop compiling data points in combined telegrams. The compiling has to be explicitly excluded by /S parameters (see Section 3.5.3).

3.2.5 Suppression of alarm messages in the event of a bus failure

If one or more slaves fail to respond, the Modbus master normally sets the Reliability parameter of all the affected I/O blocks to UNRELIABLE_OTHER. In the case of a common source, a very large number of alarms can be triggered (a "burst"), which can overload the entire system.

We recommend controlling this with the resources of XWP Desigo V4. With V4.1 and later, TX Open Modbus offers in addition the function /R15 to suppress all alarms and device faults.

3.3 Parameters for Modbus devices and communication

Information for the devices is stored in three fields:

- ThirdDevice DevAddr includes a number, the device address.
- ThirdBus BusType RS232, RS485, RS485T(with terminating resistor).
- IPv4 GlobalCommSettings: BusCommSettings or DeviceCommSettings include a string with a number of fixed parameters followed by optional supplemental information.

The data in these engineering fields are not checked when entering them into XWP and TX Open tool. Errors are recognized when starting up TX Open modules and reported to the diagnostics page of the TX Open Tool. Engineering errors (incorrectly defined points) are visible in the system in an error state.

Monitor communications:

GlobalCommSettings and DeviceCommSettings are used. Modbus is normally operated with just one type of communications that are set in the Global-CommSettings. You should only set differing DeviceCommSettings when Modbus devices with differing setting parameters are intentionally operated on the same physical cable.

The parameters are preset in the IO Open POINT template. It modifies the plant in the Point Configurator and TX Open tool. It is then transferred from Point Configurator to TX Open tool and downloaded to TX Open modules in the IOOCD.

These parameters are read after each power up and for each change to the configuration. In other words, changes during runtime take effect only after a restart of TX Open modules or after the download of a new IOOCD with the TX Open tool.

If no Communications Settings are defined, then the Bus Comm Settings are applied. Newly the parameter /SUP, /SD, /TD und /L can be used in the Bus Comm Settings.

Order of setting parameters in the fields GlobalCommSettings and DeviceCommSettings:

"Baudrate,Parity,Length,Stopbit,BusType,FrameTimeout,CharTimeout,Transfermode FreeText"

Table 3–1: Setting parameters for communication

Type	Input field	Description	Example
Device address	Modbus address	1..250	3
BusType	BusType	Modbus on RS232 or RS485, for RS485 without / with bus connection (RS232, RS485, RS485T). Modbus TCP = IPv4	RS232
Baud rate	CommSettings	Baud rates on Modbus: 600 / 1200 / 2400 / 4800 / 9600 / 19200 / 38400 and 115200	9600
Parity	CommSettings	Parity on the Modbus N: No parity; E: Even; O: Odd M: Mark parity; S: Space parity.	N
Length	CommSettings	Number of data bits per character on the Modbus 7/8.	8
StopBit	CommSettings	Number of stop bits on the Modbus 1 - 1.5 – 2.	1
FrTmo	CommSettings	Calculate automatically Frame timeout (1 ... 20000 ms) Maximum time for the transmission of a Modbus telegram. FrTmo = 0: Calculate automatically from baud rate Default value: 0	Calculated automatically
ChTmo	CommSettings	Character timeout (1 ... 20000 ms) Maximum time between two characters on the Modbus. ChTmo = 0: Calculate automatically from baud rate Default value: 0	Calculated automatically
Transfer mode:	CommSettings	Transfer mode: "RTU"; "ASCII"; "TCP"	RTU

Temporal control of the communication (V4.1 and later)

The following parameters can be used to control the sequence of communication:

/TD=Milliseconds (TelegramDelay) Time-lag between two request telegrams.

/SD=Milliseconds (ScanDelay) Time-lag after the last request telegram to a device, before a new request of all points of the device starts (time-lag between device scan cycles).

Telegram length (V4.1 and later)

/L=number limits the maximum number of bytes in the telegram from master to slave.

**Suppression of communication faults
Fault Delay (V4.1 and later)**

/SUP=n n = number of scan cycles before an error or fault will appear

Modbus TCP Properties (V6.1 and later)

TRecv: TCP receive timeout, the response timeout for IP connection.

Values between 500 and 30000 (in milliseconds) are possible.

CloseCon: The IP connection will not be permanent established.

Monitor communications:

Assumption of communications by the TX Open modules and its slave devices must be monitored as much as possible with a tool working on the communications cable.

As an alternative, you can set the "/TRACE" in the GlobalCommSettings parameters in the IOOCD. The starts the trace with the first telegram sent from the master to the Modbus.

Trace can also be started and stopped on-the-fly in the diagnostics tab – expert mode.



V4.1 and later: Activating "/TRACE" can also be done on the "Values" page. It will be displayed on both the "Values" page and the "Diagnostics" page. Do not forget to turn it off. Risk: Files are created continuously and shortens the life-cycle of persistent storage.

When the TX Open modules is communicated, the exchanged telegrams can be viewed in the TX Open tool with the following operating sequence:

- **Diagnostics** > button **Expert** > opens the Expert page with a lot of information on the device, software versions, bus state and the selected parameters.
- Buttons on the expert page, **ProtocolAnalyser** and **ProtocolLog**. Start and stop the log on this page.

3.4 Parameters for Modbus points

The information for the points is stored in 2 fields:

- ThirdConversionSet including 4 numeric values **MI1;MI2;MI3;MI4**.
- ThirdSignalType including a text with the data type and optional supplemental information.

The data in these engineering fields are not checked when entering them into XWP and TX Open tool. Errors are recognized when starting up TX Open modules and reported to the diagnostics page of the TX Open tool. Engineering errors (incorrectly defined points) are visible in the system in an error state.

Input value mapping steps

Mapping steps (see Figure 3-):

1. The Modbus register is read by data type. There are 2 types of registers: 16 bit for UINT16, SINT16 and 1 bit for BOOL. For UINT32, SINT32, FLOAT, 2 Modbus 16-bit registers are used on 2 back-to-back addresses. For DOUBLE (*V4.1 and later*), 4 Modbus 16-bit registers are used on 4 back-to-back addresses.
2. 1, 2 or 4 registers are read using Modbus function code.
3. With Mappinginfo3 and, in particular, Mappinginfo4, are placed in the bit and byte sequence required for the TX Open modules and remain in the form of the data type.
4. The system type is mapped from the data type per Table 3– and then transmitted to the Desigo automation station via island bus. For MI, the value is incremented by 1.

Output value mapping steps

Mapping steps (see Figure 3-):

1. The value (system type) is transmitted from the Desigo automation station via the island bus and mapped on data type. For MO, the value is reduced by 1.
2. This data type is placed in another bit or byte sequence using Mappinginfo3 and, in particular, Mappinginfo4 (the same number of bits).
3. One or two registers are written using Modbus function code. There are 2 types of registers: 16 bit for UINT16, SINT16 and 1 bit for BOOL. For UINT32, SINT32, FLOAT, 2 Modbus 16-bit registers are written on 2 back-to-back addresses. For DOUBLE (*V4.1 and later*), 4 Modbus 16-bit registers are written on 4 back-to-back addresses.

Note: Additional processing functions are available to the function blocks depending on the block (e.g. Slope/Intercept).

Note: The 2 registers a comprised of 4 bytes in the sequence Register 1 Byte1, Register 1 Byte2, Register 2 Byte1, Register 2 Byte2. If another byte sequence is used in the device, you must correct is using the MappingInfo4 functions.

Table 3–2: Setting parameters for the data points

Element	Description	Areas	Description
Device addresses	Modbus address	1 .. 255	Modbus slave address is used on the device and does not need to be repeated at the point.
MappingInfo1	Modbus function code	1,2,3,4,5,6,15,16	Function code with which the data point is to be transmitted. 1 : Read Single Coils 2 : Read Input Status 3 : Read Holding Register 4 : Read Input Register 5 : Write Single Coil 6 : Write Single Register 15 : Write Multiple Coils 16 : Write Multiple Registers
MappingInfo2	Modbus Data Address	0 .. 65535	The Modbus register address of the data point, that will be used in the transmitted telegram. e.g. the Holding Register reference 40001 -> Holding Register number 1 -> Holding Register address 0. For values transmitted in two or four register (32 bit or 64 bit), the register address with the lowest register address must be entered here.
MappingInfo3	Additional Functions	0 1 .. 16	0 : BitPos function disabled and functions in MappingInfo4 enabled 1..16: BitPos enabled. The value indicates the position of the bit in the Modbus register. V4.1 and later: MappingInfo4 contains the number of bits.
MappingInfo4	Format	0 .. 9	Indicates which function is to be used for data conversion. V4.1 and later : indicates the number of bits for MappingInfo3.
Data type (followed by space and free syntax)	Data type	BOOL, UINT16, SINT16, UINT32, SINT32, FLOAT V4.1 and later : DOUBLE UINT48, SINT48, UINT64, SINT64 and FLOAT16	V 4.1 and 5.0: When using the BitPos function for Holding and Input register the format BOOL is not allowed, instead the format UINT16 or UNIT32 should be to be used.
Freely definable text	Controls	Any number of characters	Freely-definable text can be used as control elements for special functions. Refer to Section 3.5 for examples.

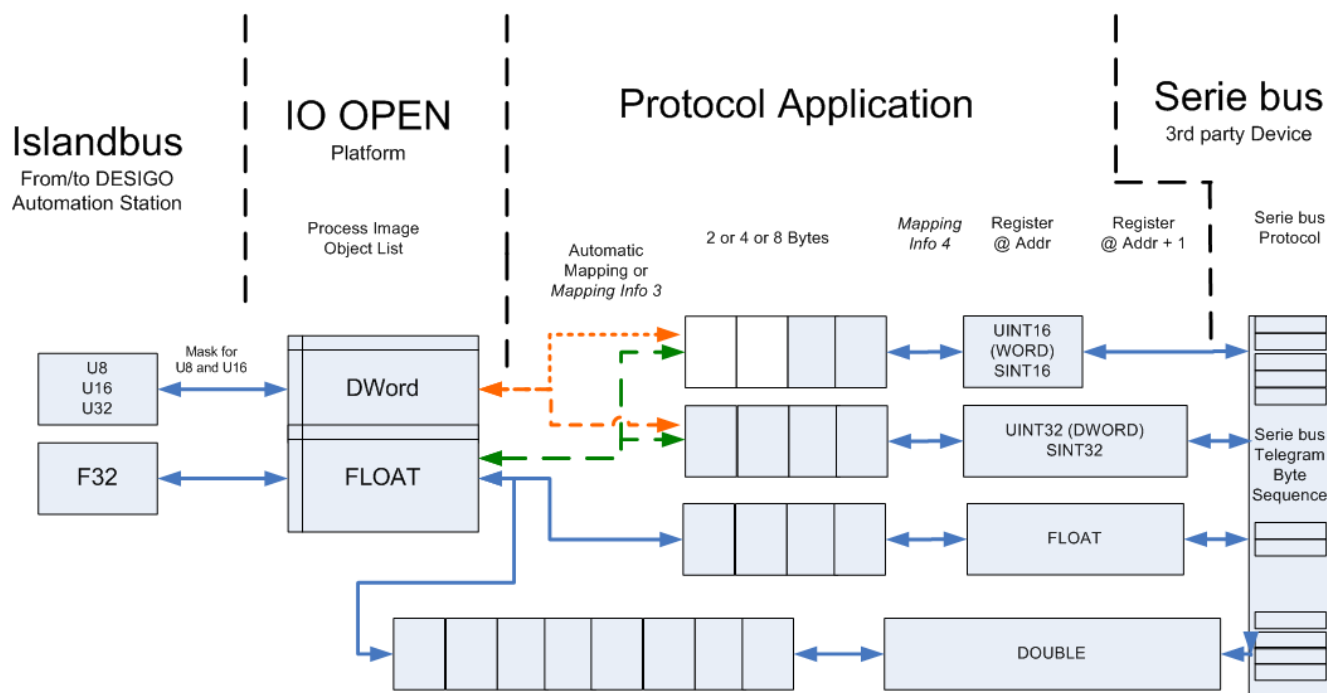
3.4.1 Data types and mapping data to function blocks

The specified data type defines the data type to which the Modbus data point is mapped. The following types are implemented: BOOL, UINT16, SINT16, UINT32, SINT32, FLOAT; 4.1 and later: DOUBLE, UINT48, SINT48, UINT64, SINT64 and FLOAT16.

**Combination of
DataType, Format,
BitPos and Function
Code.**

The format of a Modbus data point can be influenced at various points on its journey through the system. Figure 3- shows the transitions and specifies the transitions that can be defined with mapping functions to convert the data format.

Figure 3-3 Data mapping overview



3.4.2 Detailed description of the mapping information

Device information to operate as Modbus master:

The device address in master mode contains the Modbus slave address. The Modbus slave address designates the Modbus device to which the data point is to be transmitted, or from which the data point is to be read.

MappingInfo1

MappingInfo1 contains the function code used to transmit the data point on Modbus. Together with MappingInfo3, several values can be read and written efficiently.

INPUT	Input type	Output type	
01 Read Single Coil	Permitted	Prohibited	DI
02 Read Input Status	Permitted	Prohibited	DI
03 Read Holding Register	Permitted	Prohibited	AI, CI, MI
03 Read Holding Register with BitPos in MappingInfo3	Permitted	Prohibited	Read multiple DIs from one or more back-to-back registers.
04 Read Input Register	Permitted	Prohibited	AI, CI, MI
04 Read Input Register with BitPos in MappingInfo3	Permitted	Prohibited	Read multiple DIs from one or more back-to-back registers.

Function code	Operation as Master		CFC data type
OUTPUT	Input type	Output type	
05 Force Single Coil	Prohibited	Permitted	DO
06 Preset Single Register	Prohibited	Permitted	AO, MO
06 Preset Single Register with BitPos in MappingInfo3	Prohibited	Permitted	Write multiple DOIs to one or more back-to-back registers.
15 Force Multiple Coil	Prohibited	Permitted	DO
16 Preset Multiple Register	Prohibited	Permitted	AO, MO
16 Preset Multiple Register with BitPos in MappingInfo3	Prohibited	Permitted	Write multiple DOs to one or more back-to-back registers.

MappingInfo2

MappingInfo2 contains the register or bit address used to transmit the data point on Modbus (Coil/Register Numbers).

For data types with a length of 32 bits, the TX Modbus automatically uses two Modbus registers. After a data point with a length of 32 bits, therefore, the next-but-one Modbus address must be allocated to the next data point.

4.1 and later: For data types with a length of 64 bits (DOUBLE), TX Modbus automatically uses four Modbus registers. After a data point with a length of 64 bits, therefore, the next plus 4 Modbus address must be allocated to the next data point.

MappingInfo3

A value of 0 disables the MappingInfo3 function and enables the data transformation function in MappingInfo4.

The BitPos function can be enabled with values 1...16.

Activation is only permissible with the appropriate combination of data type and function code (see Table 6.3).

4.1 and later: If a BitPos is set, MappingInfo4 contains the number of bits to be used from the BitPos (bits with higher number). Example: x;x;3;4 uses bits 3, 4, 5 and 6.

Details of BitPos Functions

- DI: extracts a bit from a 16 bit register and assigns it to a DI.
- DO: inserts the value from a DO in the indicated position and writes the entire 16 bit register.

4.1 and later:

- MI: extracts the required number of bits from a 16 bit register starting from the indicated position and assigns them to an MI.
- MO: inserts the value from the MO in the indicated range of bits and writes the entire 16 bit register.

Note: If the same register (same device address, same Modbus data address) is addressed using BitPos functions, all bits are written to the register.

MappingInfo4

Values 1 ... 9 enable the data conversion functions (see Table 3–).

Activation is only permissible with the appropriate combination of data type and function code (see Table 3– and Table 3–).

If a value of 0 is defined, no data conversion will take place.

Combinations of data types

Table 3– shows how data type combinations are mapped.

- OK shows reasonable data types and their restrictions.
- The letters indicate the explanations on data conversion.

Table 3–4: Data type combinations

Desigo Resource Type	Island bus	Modbus					
		BOOL	UINT16	SNIT16	UINT32	SINT32	FLOAT / DOUBLE
AI	F32, IEEE FLOAT	OK	OK	OK	OK (A)	OK (B)	OK (G)
AO	F32, IEEE FLOAT	(H)	(K)	(K)	(K)	(K)	OK (G)
DI	U8 0=false, 1=true	OK	(L)	(L)	(L)	(L)	(L)
DO	U8 0=false, 1=true	OK	OK (C)	OK (C)	OK (C)	OK (C)	OK (C)
MI (E)	U16	OK (D)	OK	(M)	(M)	(M)	(M)
MO (F)	U16	(H)	OK	(N)	OK	OK	OK
CI	U32	OK	OK	(P)	OK	(P)	(P)

Comments:

- (A)** Loss of accuracy for numbers >16,777,215 (mantissa for FLOAT is 24 bit only).
- (B)** Loss of accuracy for numbers >16,777,215 and for numbers < -16,777,215 (mantissa for FLOAT is 24 bit only).
- (C)** Number values 0 (for false) and unequal 0 (for true) are generated.
- (D)** The Boolean value is values 0 or 1 for MO.
- (E)** For the M blocks the value range 0...n of the TX Open modules is raised to the value range 1...n+1 of the Desigo system.
4.1 and later: Furthermore the value read from the Modbus device can be corrected with /M=Correction, e.g. UINT16/M=-2.
- (F)** For the M blocks the value range 1...n+1 of the Desigo system is lowered to the value range 0...n of the TX Open modules.
4.1 and later: Furthermore the value sent to the Modbus device can be corrected with /M=Correction, e.g. UINT16/M=1.
- (G)** 4.1 and later: Loss of accuracy for DOUBLE. The DOUBLE value can be shifted into the Float range using the /D parameter.

Table 3–5: MappingInfo3 conversions

Mapping Info 3	Functions	Description
H	The entire FLOAT or multistate range is mapped to a bit (coil).	If the output value (from AO, MO) is exactly 0, the bit becomes 0, else 1.
K	FLOAT is mapped to a smaller, integer range. See table "Number ranges" for limits.	Values out of range are mapped to the next value within the range. The value is rounded to the next integer number.
L	A logical value DI (U8) 0=false / 1=true is generated from any number.	If the number is zero, DI also becomes 0; if not 0, then DI becomes 1.
M	A number is mapped to multistate (with smaller number range). See table "Number ranges" for limits.	Values out of range are mapped to the next value within the range. The value is rounded to the next integer number.
N	Multistate is mapped to the smaller number range of SINT16.	If multistate is greater than 32,767, then 32,767 is written to SINT. PS: Alternative, see Function 20.
P	A number is mapped to a counter (integer >=0) with a different number range. See table "Number ranges" for limits.	The value is rounded to the next integer number. Values out of range are mapped to the next value within the range. In particular: Negative numbers cannot be mapped, thus become 0. FLOAT numbers >=4,294,967,295 become 4,294,967,295.

Loss of accuracy and range

The mapping causes loss of accuracy and range. The following information on the value ranges allows for estimating the losses and assessing if the loss is relevant for the value and acceptable.

Table 3–6: Number ranges

Type	Lower limit	Upper limit	Comment
DI, DI, U8, BOOL	0	1	Mapping of numbers: (NOT 0) to 1.
MI, MO, U16, UINT16	0	65'335	0 to $2^{16} - 1$
SINT16	-32'768	32'767	-2^{15} to $2^{15} - 1$
CI, U32, UINT32	0	4'294'967'295	0 to $2^{32}-1$
SINT32	-2'147'483'648	2'147'483'647	-2^{31} to $2^{31}-1$
FLOAT	$-(1-2^{-24}) \times 2^{128} \approx -3,403 \cdot 10^{38}$	$(1-2^{-24}) \times 2^{128} \approx 3,403 \cdot 10^{38}$	Exponent: 2^{-126} to 2^{127} Smallest number: $2^{-23} \times 2^{-126} = 2^{-149} \approx 1.401 \cdot 10^{-45}$.
DOUBLE	$(1-2^{-53}) \times 2^{1023} \approx 1.79 \cdot 10^{308}$	$(1-2^{-53}) \times 2^{1023} \approx 1.79 \cdot 10^{308}$	Exponent: 2^{-1022} to 2^{1023} Smallest number: $2^{-53} \times 2^{-1022} \approx 5 \cdot 10^{-324}$

Loss of accuracy with conversions into FLOAT

The conversion of 32-bit data types into FLOAT values is associated with a loss of accuracy. The loss is 8 or 9 bits (for a signed and unsigned 32-bit value respectively). **This type of conversion arises with the combination of standard input blocks with data types SINT32 and UINT32.**

FLOAT:

- Accuracy 7 decimal places (23 bit mantissa, i.e. accuracy $2^{24} = 16,777,216$).
- 6 bit exponent, 1 bit sign.
- Smallest possible number: $2^{-23} \times 2^{-126} = 2^{-149} \approx 1.401 \cdot 10^{-45}$

Double:

- Accuracy (Double precision) approx. 20 decimal places (53 Bit mantissa, i.e. Accuracy 2^{53})
- 11 Bit exponent, 1 Bit sign
- This high accuracy and high range of numbers are not supported in the Designo system. The values are converted to FLOAT (AI) or UINT32 (CI). Prior to conversion, a value can be shifted to the range of the AI (FLOAT) or CI (UINT32) using the /D parameter (Division).



Note!

If these combinations also include data conversion codes 1 with MappingInfo 4 (Swap Register) or 5 (BCD32B) the loss of accuracy moves to the most significant bits!

The available conversion functions are as follows as MappingInfo4:

Table 3–7: MappingInfo4 conversion functions

Mapping Info4 Format Parameter	Name	Function	Possible with the data types												
0	No conversion	No MappingInfo4 conversion	All												
1	SwapRegisters	The two 16 Bit Registers are swapped before sending to the Modbus and after receiving from the Modbus	UINT32 / SINT32 / FLOAT												
2	BCD16A	The received data from the Modbus will be interpreted as a 16 Bit BCD Code and transformed to a binary value. A (binary) value to be sent on the Modbus will be transformed to 16 Bit BCD Code A : The first byte in the telegram is the most significant.	UINT16												
	Example:	<table border="1"> <tr> <td>Modbus Byte1</td> <td>Modbus Byte2</td> </tr> <tr> <td>Hex: 0x47</td> <td>Hex: 0x19</td> </tr> <tr> <td>BCD: 4 7</td> <td>BCD: 1 9</td> </tr> </table>	Modbus Byte1	Modbus Byte2	Hex: 0x47	Hex: 0x19	BCD: 4 7	BCD: 1 9							
Modbus Byte1	Modbus Byte2														
Hex: 0x47	Hex: 0x19														
BCD: 4 7	BCD: 1 9														
		Value = 4719													
3	BCD16B	The received data from the Modbus will be interpreted as a 16 Bit BCD Code and transformed to a binary value. A (binary) value to be sent on the Modbus will be transformed to 16 Bit BCD Code B : The first byte in the telegram is the least significant.	UINT16												
	Example:	<table border="1"> <tr> <td>Modbus Byte1</td> <td>Modbus Byte2</td> </tr> <tr> <td>Hex: 0x19</td> <td>Hex: 0x47</td> </tr> <tr> <td>BCD: 1 9</td> <td>BCD 4 7</td> </tr> </table>	Modbus Byte1	Modbus Byte2	Hex: 0x19	Hex: 0x47	BCD: 1 9	BCD 4 7							
Modbus Byte1	Modbus Byte2														
Hex: 0x19	Hex: 0x47														
BCD: 1 9	BCD 4 7														
		Value = 4719													
4	BCD32A	The received data from the Modbus will be interpreted as a 32 Bit BCD Code and transformed to a binary value. A (binary) value to be sent on the Modbus will be transformed to 32 Bit BCD Code A: The first byte in the telegram is the most significant	UINT32												
	Example:	<table border="1"> <tr> <td>MB Byte1</td> <td>MB Byte2</td> <td>MB Byte3</td> <td>MB Byte4</td> </tr> <tr> <td>Hex: 0x53</td> <td>Hex: 0x34</td> <td>Hex: 0x47</td> <td>Hex: 0x19</td> </tr> <tr> <td>BCD:5 3</td> <td>BCD: 3 4</td> <td>BCD: 4 7</td> <td>BCD: 1 9</td> </tr> </table>	MB Byte1	MB Byte2	MB Byte3	MB Byte4	Hex: 0x53	Hex: 0x34	Hex: 0x47	Hex: 0x19	BCD:5 3	BCD: 3 4	BCD: 4 7	BCD: 1 9	
MB Byte1	MB Byte2	MB Byte3	MB Byte4												
Hex: 0x53	Hex: 0x34	Hex: 0x47	Hex: 0x19												
BCD:5 3	BCD: 3 4	BCD: 4 7	BCD: 1 9												
		Value = 53344719													
5	BCD32B	The received data from the Modbus will be interpreted as a 32 Bit BCD Code and transformed to a binary value. A (binary) value to be sent on the Modbus will be transformed to 32 Bit BCD Code B: The first byte in the telegram is the least significant.	UINT32												
	Example:	<table border="1"> <tr> <td>MB Byte1</td> <td>MB Byte2</td> <td>MB Byte3</td> <td>MB Byte4</td> </tr> <tr> <td>Hex: 0x19</td> <td>Hex: 0x47</td> <td>Hex: 0x34</td> <td>Hex: 0x53</td> </tr> <tr> <td>BCD: 1 9</td> <td>BCD: 4 7</td> <td>BCD: 3 4</td> <td>BCD: 5 3</td> </tr> </table>	MB Byte1	MB Byte2	MB Byte3	MB Byte4	Hex: 0x19	Hex: 0x47	Hex: 0x34	Hex: 0x53	BCD: 1 9	BCD: 4 7	BCD: 3 4	BCD: 5 3	
MB Byte1	MB Byte2	MB Byte3	MB Byte4												
Hex: 0x19	Hex: 0x47	Hex: 0x34	Hex: 0x53												
BCD: 1 9	BCD: 4 7	BCD: 3 4	BCD: 5 3												
		Value = 53344719													
6	ACM16	The received data from the Modbus will be interpreted as follows. The value to be sent on the Modbus will be transformed as follows. Value =Low Register + (High Register * 1000) The first register transmitted in the telegram is the Low Register	UINT32												
	Example:	<table border="1"> <tr> <td>Byte 1</td> <td>Byte 2</td> <td>Byte 3</td> <td>Byte 4</td> </tr> <tr> <td>Hex: 0x02</td> <td>Hex: 0xD6</td> <td>Hex: 0x02</td> <td>Hex: 0x83</td> </tr> <tr> <td colspan="2">726</td> <td colspan="2">643</td> </tr> </table>	Byte 1	Byte 2	Byte 3	Byte 4	Hex: 0x02	Hex: 0xD6	Hex: 0x02	Hex: 0x83	726		643		
Byte 1	Byte 2	Byte 3	Byte 4												
Hex: 0x02	Hex: 0xD6	Hex: 0x02	Hex: 0x83												
726		643													
		Value = 726 = 643726 + (643 * 1000)													
7	U32BITMOD10000	The received data from the Modbus will be interpreted as follows. The value to be sent on the Modbus will be transformed as follows. Value = (High Register * 10000) + Low Register The first register transmitted in the telegram is the High Register	UINT32												
	Example:	<table border="1"> <tr> <td>Byte 1</td> <td>Byte 2</td> <td>Byte 3</td> <td>Byte 4</td> </tr> <tr> <td>Hex: 0x04</td> <td>Hex: 0xD2</td> <td>Hex: 0x16</td> <td>Hex: 0x2E</td> </tr> <tr> <td colspan="2">1234</td> <td colspan="2">5678</td> </tr> </table>	Byte 1	Byte 2	Byte 3	Byte 4	Hex: 0x04	Hex: 0xD2	Hex: 0x16	Hex: 0x2E	1234		5678		
Byte 1	Byte 2	Byte 3	Byte 4												
Hex: 0x04	Hex: 0xD2	Hex: 0x16	Hex: 0x2E												
1234		5678													
		Value = (1234 * 10000) + 5678 = 12345678													

Mapping Info4 Format Parameter	Name	Function	Possible with the data types																
8	S32BITMOD10000	The received data from the Modbus will be interpreted as follows. The value to be sent on the Modbus will be transformed as follows. Value = (High Register * 10000) + Low Register The first register transmitted in the telegram is the High Register	SINT32																
	Example:	<table border="1"> <tr> <td>MB Byte1</td> <td>MB Byte2</td> <td>MB Byte3</td> <td>MB Byte4</td> </tr> <tr> <td>Hex: 0xFB</td> <td>Hex: 0x2E</td> <td>Hex: 0xE9</td> <td>Hex: 0xD2</td> </tr> <tr> <td colspan="2" style="text-align: center;">-1234</td> <td colspan="2" style="text-align: center;">-5678</td> </tr> <tr> <td colspan="4" style="text-align: center;">Value = (-1234 * 10000) + -5678 = -12345678</td> </tr> </table>	MB Byte1	MB Byte2	MB Byte3	MB Byte4	Hex: 0xFB	Hex: 0x2E	Hex: 0xE9	Hex: 0xD2	-1234		-5678		Value = (-1234 * 10000) + -5678 = -12345678				
MB Byte1	MB Byte2	MB Byte3	MB Byte4																
Hex: 0xFB	Hex: 0x2E	Hex: 0xE9	Hex: 0xD2																
-1234		-5678																	
Value = (-1234 * 10000) + -5678 = -12345678																			
9	SwapBytes	The Bytes of the registers are swapped before transmission to the Modbus and after receipt from the Modbus	All																
	Example:	<table border="1"> <tr> <td>MB Byte1</td> <td>MB Byte2</td> <td>MB Byte3</td> <td>MB Byte4</td> </tr> <tr> <td>Hex: 0x7B</td> <td>Hex: 0x14</td> <td>Hex: 0x48</td> <td>Hex: 0x42</td> </tr> <tr> <td colspan="4" style="text-align: center;">Value = 50.02</td> </tr> </table>	MB Byte1	MB Byte2	MB Byte3	MB Byte4	Hex: 0x7B	Hex: 0x14	Hex: 0x48	Hex: 0x42	Value = 50.02								
MB Byte1	MB Byte2	MB Byte3	MB Byte4																
Hex: 0x7B	Hex: 0x14	Hex: 0x48	Hex: 0x42																
Value = 50.02																			



Note!

**Not all combinations of function code and MappingInfo4 data conversions are meaningful.
The following tables describe allowed functions.**

Table 3–8: Allowed MappingInfo4 functions for input

Data Type	BOOL	SINT16	UINT16	SINT32 (2 Registers)	UINT32 (2 Registers)	FLOAT (2 Registers)
Mapping Info4 Format Parameter						
0 No Conv.	01 Read Coil 02 Read Input	01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	03 Read Holding 04 Read Input
1 Swap Register				01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	03 Read Holding 04 Read Input
2 BCD16A			01 Read Coil 02 Read Input 03 Read Holding 04 Read Input			
3 BCD 16B			01 Read Coil 02 Read Input 03 Read Holding 04 Read Input			
4 BCD32A					01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	
5 BCD32B <i>Nicht verwenden mit Float (Standard I/O)</i>					01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	
6 ACM16					01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	
7 U32BITMOD10000					01 Read Coil 02 Read Input 03 Read Holding 04 Read Input	
8 S32BITMOD10000				01 Read Coil 02 Read Input 03 Read Holding 04 Read Input		
9 SwapBytes	01 Read Coil 02 Read Input	01 Read Coil 02 Read Input	01 Read Coil 02 Read Input	01 Read Coil 02 Read Input	01 Read Coil 02 Read Input	(also Double)
		03 Read Holding 04 Read Input	03 Read Holding 04 Read Input	03 Read Holding 04 Read Input	03 Read Holding 04 Read Input	03 Read Holding 04 Read Input
Number of bits (see MappingInfo3)	03 Read Holding 04 Read Input					

Table 3–9: Allowed MappingInfo4 functions for output

Mapping Info DataType	BOOL	SINT16	UINT16	SINT32 (2 Registers)	UINT32 (2 Registers)	FLOAT (2 Registers)
Mapping Info4 Format Parameter						
0 No Conv.	05 Force S. Coil 06 Preset S.Reg 15 Force M.Coil 16 Preset M.Reg	06 Preset S.Reg 16 Preset M.Reg	06 Preset S.Reg 16 Preset M.Reg	16 Preset M.Reg	16 Preset M.Reg	16 Preset M.Reg
1 Swap Register				16 Preset M.Reg	16 Preset M.Reg	16 Preset M.Reg
2 BCD16A			06 Preset S.Reg 16 Preset M.Reg			
3 BCD 16B			06 Preset S.Reg 16 Preset M.Reg			
4 BCD32A					16 Preset M.Reg	
5 BCD32B					16 Preset M.Reg	
6 ACM16					16 Preset M.Reg	
7 U32BITMOD10000					16 Preset M.Reg	
8 S32BITMOD10000				16 Preset M.Reg		
9 SwapBytes	05 Force S. Coil 06 Preset S.Reg 15 Force M.Coil 16 Preset M.Reg	06 Preset S.Reg 16 Preset M.Reg	06 Preset S.Reg 16 Preset M.Reg	16 Preset M.Reg	16 Preset M.Reg	16 Preset M.Reg
Number of bits (see MappingInfo3)	06 Preset S.Reg 16 Preset M.Reg					

3.5 Extensions to the Modbus protocol

In many devices equipped with Modbus, the Modbus protocol is not fully implemented. Not all the function codes can be used, and the data points are not addressed strictly in accordance with the Modbus protocol.

In order to ensure the efficient integration of these devices, the TX Modbus solution provides the add-ons described below.

3.5.1 Transmit 32-bit and 64-bit values

The Modbus protocol defines the 16-bit register as the largest data unit. 32-bit values must therefore be transmitted in two registers and 64-bit values in four registers.

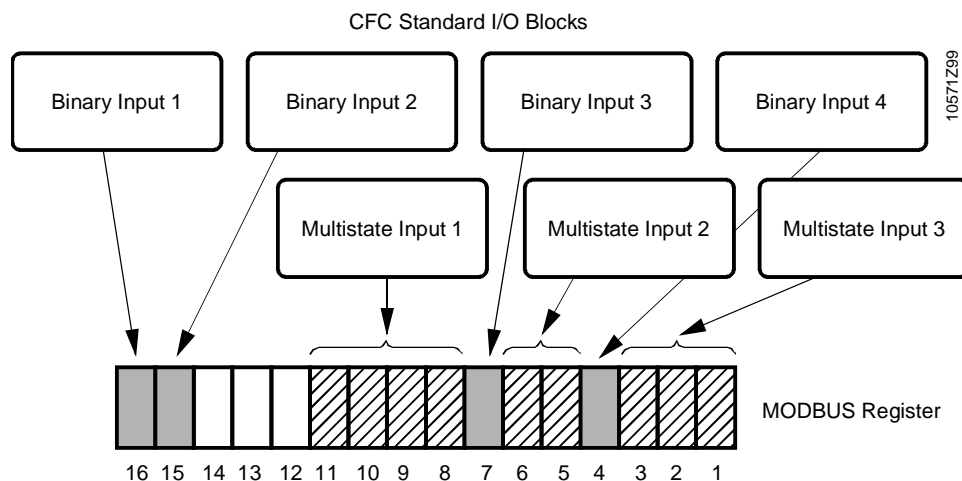
The Modbus protocol application automatically occupies two / four 16-bit registers for the data types concerned (FLOAT, UINT32 and SINT32, DOUBLE UINT48, SINT48, UINT64, SINT64 and FLOAT16), and also ensures that they are always combined for transmission in a shared telegram (note that this requires a function code (3, 4, 16) capable of transmitting several registers in a single telegram.

3.5.2 BitPos function: Read/Write individual bits and multiple bits in the Modbus registers

The Modbus protocol only allows single bits to be read with Function Codes 01 (Read Coil Status), 05 (Force Single Coil) and 15 (Force Multiple Coils).

However, many Modbus devices do not support all these function codes. The BitPos function makes it possible to handle single bits despite this fact, and without additional work in the CFC program.

BitPos can be used to link the I/O block types "Binary Input" and "Binary Output" with single bits of a 16-bit Modbus register.



Example of a Modbus register linked with Function Code 04 (Read Input Register)

Indicating the number of bits in MappingInfo4 allows you to read multiple bits from a register. The rest of the bits of the register are dismissed. This makes especially sense for MI and MO, where parts of registers belong together, e.g. an error code. Bits or a register not adjacent to each other must be treated via multiple MIs and MOs.

Note

If different bit positions are accessed within the **same register number**, the **same Modbus function** must be used to write all bits from this register.

Examples:

- | | |
|--|--------------------|
| ThirdConversionSet="4;100;1;3" , ThirdSignalType="UINT16" | Multistate Input 3 |
| ThirdConversionSet="4;100;4;1" , ThirdSignalType="UINT16" | Binary Input 4 |
| ThirdConversionSet="4;100;5;2" , ThirdSignalType="UINT16" | Multistate Input 2 |
| ThirdConversionSet="4;100;7;1" , ThirdSignalType="UINT16" | Binary Input 3 |
| ThirdConversionSet="4;100;8;4" , ThirdSignalType="UINT16" | Multistate Input 1 |
| ThirdConversionSet="4;100;15;1" , ThirdSignalType="UINT16" | Binary Input 2 |
| ThirdConversionSet="4;100;16;1" , ThirdSignalType="UINT16" | Binary Input 1 |

Bit collecting function

With this way of addressing a BI can also be the result of multiple bits. BI = 0 if no bit is set, BI = 1 if one or several bits are set (example: alarm collection).

3.5.3 Suppress transmission of a variable number of data points

Some simple devices may not support the transmission of several data points in a combined Modbus telegram.

In such cases the process of combining data points can be suppressed for specific individual data points. The "free syntax" in the IO address string is used for this purpose. The parameter is /S (suppress).

Example: `ThirdSignalType="UINT16 /S"`

A data point defined in this way is always transmitted in a separate telegram.

3.5.4 Feedback Function and value sequences

For the transmission of pulses or specific sequences of values of a Modbus data point, it is important to ensure that a given state or value has been transmitted before writing the next state or value to the data point. To be able to easily program such mechanisms in the CFC of a Modbus master, the Modbus protocol application provides for a mechanism supplying the feedback value even if the device does not support a feedback value.

As soon as the value is written and the slave acknowledges receipt, the value is written to the feedback value by the Modbus protocol application. The last value acknowledged as having been received by the Modbus slave then appears as the feedback value.

In Output blocks, the block address and the feedback value are entered in the feedback address, and both addresses are labeled with /F in IOOPT and IOOCD.

Example:

```
AO ThirdConversionSet="6;22;0;0" ThirdSignalType="UINT16 /F"  
AI ThirdConversionSet="6;22;0;0" ThirdSignalType="UINT16 /F"
```

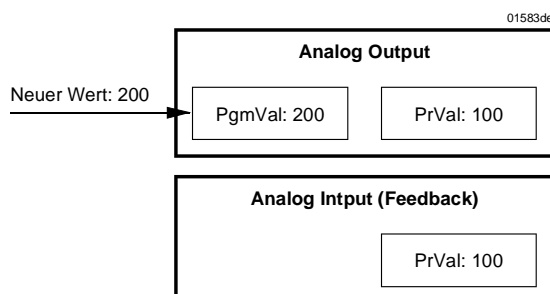
Feedback function /F

Feedback function /F is applicable for Out data point with the corresponding feedback In data point.

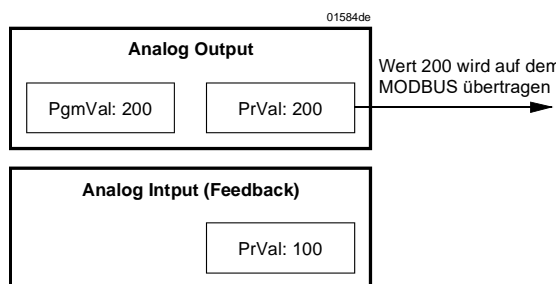
Example

Modbus master with a data point which contains a value.
This value is to be written to the slave.

A) A new value of 200 is written to the analog output

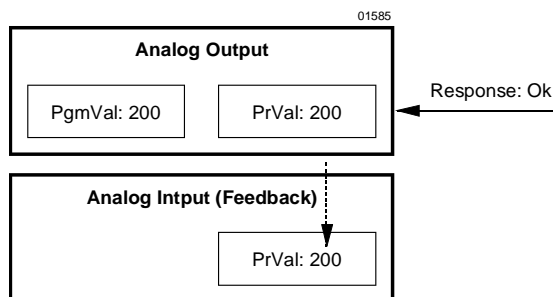


B) The new value is transmitted to the slave via the Modbus



C) Receipt of the value is acknowledged by the slave.

PresentValue is then updated in FeedbackValue.



3.5.5 Periodic writing of outputs

There are three levels determining frequency of writing.

To activate periodic writing of outputs, the corresponding R-functions of free syntax must be appended to the mapping strings.

R function	Write interval
/R1	Every minute
/R2	Every hour
/R3	Every 10 hours
/R4	Every 24 hours

- Write processes of various levels must not necessarily be mutually synchronized.
- A data point cannot simultaneously be assigned to several intervals.

Examples for mapping strings

Write Analog Output every hour:

```
ThirdConversionSet="16;22;0;0" ThirdSignalType="FLOAT /R2"
```

Write analog output every 10 hours:

```
ThirdConversionSet="16;44;0;0" ThirdSignalType="FLOAT /R3 "
```

3.5.6 Format conversion and value shifting

In many devices incorporating the Modbus, special data formats (e.g. BCD, ACM etc.) are used. There is a further problem, in that values which occupy two registers, the sequence of the actual values can be defined in different ways.

The Modbus protocol application provides a number of conversion functions as described in Section 3.4.2.

V4.1 and later: With MI and MO parameter /M=Correction allows you to correct the value.

Examples: UINT16/M=7
 UINT16/M=-1

V4.1 and later: With FLOAT and DOUBLE, parameter /D=Divisor allows you to shift the value read from the device into the admissible range. The shifted value is visible in the "Values" window and is transmitted to Desigo in this form.

Examples: FLOAT/D=100 divides the value read from the device by 100.
 DOUBLE/D=1000 divides the value read from the device by 1000.
 DOUBLE/D=0,01 multiplies the value read from the device by 100.

3.5.7 Grouping of output registers

Output registers, which must be in a sequence without interruption, can be grouped. This allows writing data, which are distributed over several registers, in a single write command when at least one of these writing data has been written. The grouping function can be used only with **Force Multiple Coil (Function code = 15)** or **Preset Multiple Register (Function code = 16)**.

/GN Designates the start of the data to be grouped.
/G Designates the following data to be grouped.

3.5.8 BACnet reliability

The status of the data points is mapped to the parameter [Rlb] (Reliability). In the standard I/O blocks, an Rlb parameter is available for each data point.

Reliability value	Meaning in the master
NO_FAULT_DETECTED	The connection between data point and slave is OK.
PROCESS_ERROR	Setting parameter error.
UNRELIABLE_OTHER	The slave responds with a negative reply ID.
INVALID_VALUE	Error during data conversion.

Additional information can be obtained using TX Open tool (diagnostics) in the event of an error.

Suppressing alarm messages in the event of a bus failure

If one or more slaves fail to respond, the Modbus master normally sets the Reliability parameter of all the affected I/O blocks to UNRELIABLE_OTHER. In the case of a common source, a very large number of alarms can be triggered (a "burst"), which can overload the entire system.

This burst can be prevented if an alarm reference point is selected for each device and all other points are mapped to simple system objects without alarm handling.

3.5.9 Broadcast telegrams

Some third-party devices are capable of handling broadcast telegrams.

The latest protocol application (>V6.00.660) supports this function.

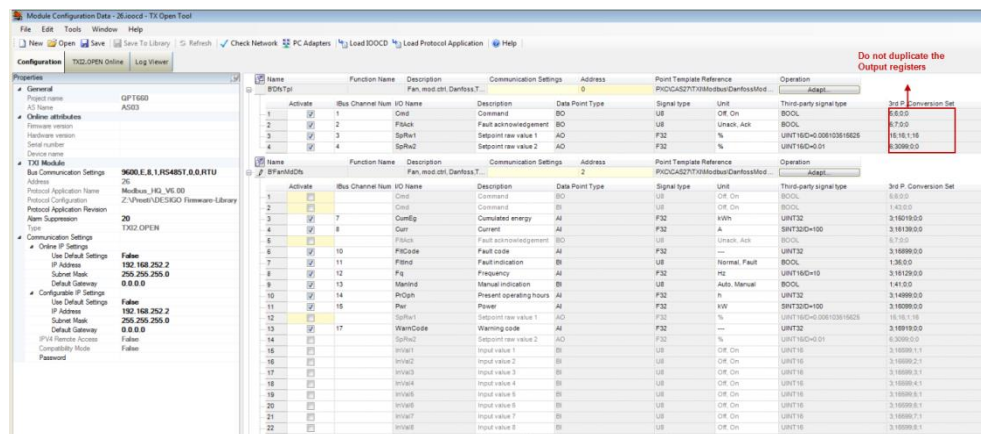
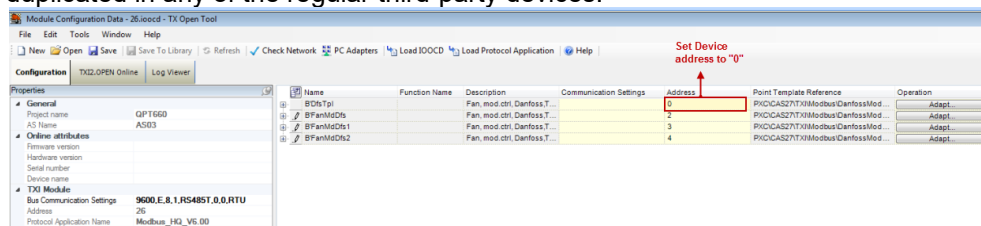
Sending broadcast telegrams is useful for Output data points only.

For the configuration it is required to add an additional Device with address = 0.

Data points configured underneath this device do not expect a response from the slaves.

Change of value for the output data points is updated on priority

Note: Register addresses which are used for the broadcast should not be duplicated in any of the regular third-party devices.



4 Create protocol applications for projects (by an RC)

HQ offers advanced training. Participants must have solid knowledge of C++ programming and basic knowledge of Desigo. The course provides the necessary tools and know-how to create new protocol applications. This specifically includes development environment, programming guides, interface descriptions, source samples, and template samples.

Hosting and course prices by agreement.

Note: Normally, no programming is required to define additional devices in existing protocol applications. It is enough to create a template. Examples of templates for Modbus are provided by the library. You can edit the templates in the TX Open tool or use them to create new templates.

5 Technical data

For technical data, refer to the following documents:

Type	Device	Data sheet
PXC50/100/200	System controller with island bus communication	CM1N9222
TXIx.OPEN	TX Open RS232/485 module	CM1N8187
	TX Open tool (online help)	--
	Point Configurator in XWP (online help)	--

6 SW and FW versions

See Release Notes CM2N8187en.

Index

3	
32-bit values.....	33
A	
Alarm suppression	20
Application	16
ASCII mode.....	17
B	
BACnet reliability.....	38
BitPos.....	34
BitPos function	34
D	
Data conversion	25
Diagnostics	8
F	
Format conversion	37
Function codes.....	17
L	
Loss of accuracy	29, 31
M	
Mapping information	26
MappingInfo1.....	26
Master / Slave protocol.....	17
Modbus master	
Poll response	18
Modbus protocol	33
Modbus telegram.....	19
P	
Poll response.....	18
Poll telegrams.....	19
R	
Reliability Rlb.....	38
Remote Terminal Unit (RTU)	17
T	
Transmission mode	17
TX Open tool	23
U	
Update output values.....	19
X	
XWP.....	23